PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F — A2

(11) International Publication Number: WO 96/42041

(43) International Publication Date: 27 December 1996 (27.12.96)

(21) International Application Number: PCT/US96/07838

(22) International Filing Date: 3 June 1996 (03.06.96)

(30) Priority Data:

08/474,096	7 June 1995 (07.06.95)	US
08/486,797	7 June 1995 (07.06.95)	US

(81) Designated States: AU, CA, DE, GB, IL, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

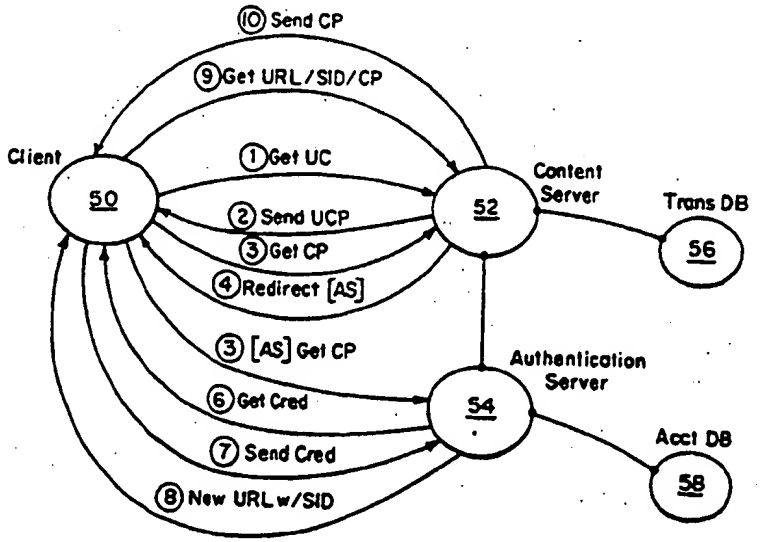
Published
Without international search report and to be republished upon receipt of that report.

(71) Applicant: OPEN MARKET, INC. [US/US]; 215 First Street, Cambridge, MA 02142 (US).

(72) Inventors: LEVERGOOD, Thomas, Mark; 9 North Street, Hopkinton, MA 01748 (US). STEWART, Lawrence, C.; 1 Arborwood Drive, Burlington, MA 01803 (US). MORRIS, Stephen, Jeffrey; 3 Kings Pine Road, Westford, MA 01886 (US). PAYNE, Andrew, C.; 5 Lewis Street, Lincoln, MA 01773 (US). TREESE, George, Winfield; 81 Saco Street, Newton, MA 02164 (US). [illegible]D, David, K.; 26 Pigeon Hill Road, Weston, MA [illegible] (US).

(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, Two Militia Drive, Lexington, MA 02173 (US).

(54) Title: INTERNET SERVER A[illegible]



(57) Abstract

This invention relates to methods for controlling and monitoring access to network servers. In particular, the process described in the invention includes client-server sessions over the Internet involving hypertext files. In the hypertext environment, a client views a document transmitted by a content server with a standard program known as the browser. Each hypertext document or page contains links to other hypertext pages which the user may select to traverse. When the user selects a link that is directed to an access-controlled file, the server subjects the request to a secondary server which determines whether the client has an authorization or valid account. Upon such verification, the user is provided with a session identification which allows the user to access to the requested file as well as any other files within the present protection domain.

INTERNET SERVER ACCESS CONTROL AND MONITORING SYSTEMS

Reference to Appendix

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by any one of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

Background of the Invention

The Internet, which started in the late 1960s, is a vast computer network consisting of many smaller networks that span the entire globe. The Internet has grown exponentially, and millions of users ranging from individuals to corporations now use permanent and dial-up connections to use the Internet on a daily basis worldwide. The computers or networks of computers connected to the Internet, known as "hosts", allow public access to databases featuring information in nearly every field of expertise and are supported by entities ranging from universities and government to many commercial organizations.

The information on the Internet is made available to the public through "servers". A server is a system running on an Internet host for making available files or documents contained within that host. Such files are typically stored on magnetic storage devices, such as tape drives or fixed disks, local to the host. An Internet server may distribute information to any computer that requests the files on a host. The computer making such a request is known as the "client", which may be an Internet-connected

Created in 1991, the Web is based on the concept of "hypertext" and a transfer method known as "HTTP" (Hypertext Transfer Protocol). HTTP is designed to run primarily over TCP/IP and uses the standard Internet setup, where a server issues the data and a client displays or processes it. One format for information transfer is to create documents using Hypertext Markup Language (HTML). HTML pages are made up of standard text as well as formatting codes which indicate how the page should be displayed. The Web client, a browser, reads these codes in order to display the page. The hypertext conventions and related functions of the world wide web are described in the appendices of U.S. Patent Application Serial No. 08/328,133, filed on October 24, 1994, by Payne et al. which is incorporated herein by reference.

Each Web page may contain pictures and sounds in addition to text. Hidden behind certain text, pictures or sounds are connections, known as "hypertext links" ("links"), to other pages within the same server or even on other computers within the Internet. For example, links may be visually displayed as words or phrases that may be underlined or displayed in a second color. Each link is directed to a web page by using a special name called a URL (Uniform Resource Locator). URLs enable a Web browser to go directly to any file held on any Web server. A user may also specify a known URL by writing it directly into the command line on a Web page to jump to another Web page.

The URL naming system consists of three parts: the transfer format, the host name of the machine that holds the file, and the path to the file. An example of a URL may b :

http://www.college.univ.edu/Adir/Bdir/Cdir/page.html,

private local area networks (LAN), system administrators implement various data-flow control mechanisms, such as the Internet "firewalls", within their networks. An Internet firewall allows a user to reach the Internet anonymously while preventing intruders of the outside world from accessing the user's LAN.

Summary of the Invention

The present invention relates to methods of processing service requests from a client to a server through a network. In particular the present invention is applicable to processing client requests in an HTTP (Hypertext Transfer Protocol) environment, such as the World-Wide Web (Web). One aspect of the invention involves forwarding a service request from the client to the server and appending a session identification (SID) to the request and to subsequent service requests from the client to the server within a session of requests. In a preferred embodiment, the present method involves returning the SID from the server to the client upon an initial service request made by the client. A valid SID may include an authorization identifier to allow a user to access controlled files.

In a preferred embodiment, a client request is made with a Uniform Resource Locator (URL) from a Web browser. Where a client request is directed to a controlled file without an SID, the Internet server subjects the client to an authorization routine prior to issuing the SID, the SID being protected from forgery. A content server initiates the authorization routine by redirecting the client's request to an authentication server which may be at a different host. Upon receiving a redirected request, the authentication server returns a response to interrogate the client and then issues an SID to a qualified client. For a new client, the authentication server may open a new account and issue an SID thereafter. A valid SID typically

relative link points to a controlled page in a different protection domain, the SID is no longer valid, and the client is automatically redirected to forward the rewritten URL to the authentication server to update the SID. The updated or new SID provides access to the new domain if the user is qualified.

The user may also elect to traverse a link to a document in a different path. This is called an "absolute link". In generating a new absolute link, the SID is overwritten by the browser. In the preferred embodiment, the content server, in each serving of a controlled Web page within the domain, filters the page to include the current SID in each absolute URL on the page. Hence, when the user elects to traverse an absolute link, the browser is facilitated with an authenticated URL which is directed with its SID to a page in a different path. In another embodiment, the content server may forego the filtering procedure as above-described and redirect an absolute URL to the authentication server for an update.

An absolute link may also be directed to a controlled file in a different domain. Again, such a request is redirected to the authentication server for processing of a new SID. An absolute link directed to an uncontrolled file is accorded an immediate access.

In another embodiment, a server access control may be maintained by programming the client browser to store an SID or a similar tag for use in each URL call to that particular server. This embodiment, however, requires a special browser which can handle such communications and is generally not suitable for the standard browser format common to the Web.

Another aspect of the invention is to monitor the frequency and duration of access to various pages both controlled and uncontrolled. A transaction log within a content server keeps a history of each client access to a

Detailed Description of the Invention:

Referring now to the drawings, Figure 1 is a graphical illustration of the Internet. The Internet 10 is a network of millions of interconnected computers 12 including systems owned by Internet providers 16 and information systems (BBS) 20 such as Compuserve or America Online. Individual or corporate users may establish connections to the Internet in several ways. A user on a home PC 14 may purchase an account through the Internet provider 16. Using a modem 22, the PC user can dial up the Internet provider to connect to a high speed modem 24 which, in turn, provides a full service connection to the Internet. A user 18 may also make a somewhat limited connection to the Internet through a BBS 20 that provides an Internet [illegible]y connection to its customers.

Figure 2A is a flowchart detailing the preferred process of the present invention and Figure 4 illustrates a sample Web page displayed at a client by a browser. The page includes text 404 which includes underlined link text 412. The title bar 408 and URL bar 402 display the title and URL of the current web page, respectively. As shown in Figure 4, the title of the page is "Content Home Page" and the corresponding URL is "http://content.com/homepage". When a cursor 414 is positioned over link text 412b, the page which would be retrieved by clicking a mouse is typically identified in a status bar 406 which shows the URL for that link. In this example the status bar 406 shows that the URL for the pointed link 412b is directed to a page called "advertisement" in a commercial content server called "content". By clicking on the link text, the user causes the browser to generate a URL GET request at 100 in Figur 2A. The browser forwards the request to a content server 120, which processes the request by first determining whether the requested page is a controlled document 102. If the request is directed to an

SID is no longer valid and, again, the user is redirected to the authentication server 122.

If the request is for a controlled page within the current domain, the content server proceeds to log the request URL, tagged with SID, and the user IP address in the transaction log 108. The content server then validates the SID 110. Such validation includes the following list of checks: (1) the SID's digital signature is compared against the digital signature computed from the remaining items in the SID and the user IP address using the secret key shared by the authentication and content servers; (2) the domain field of the SID is checked to verify that it is within the domain authorized; and (3) the EXP field of the SID is checked to verify that it is later than the current time.

If the validation passes, the content server searches the page to be forwarded for any absolute URL links contained therein 112, that is, any links directed to controlled documents in different content servers. The content server augments each absolute URL with the current SID to facilitate authenticated accesses across multiple content servers. The requested page as processed is then transmitted to the client browser for display 117. The user viewing the requested Web page may elect to traverse any link on that page to trigger the entire sequence again 100.

Figure 2B describes the details of the authentication process. The content server may redirect the client to an authentication server. The REDIRECT URL might be: "http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report". That URL requests authentication and specifies the domain and the initial URL. In response to the REDIRECT, the client browser automatically sends a GET request with the provided URL.

222 is transmitted to the client browser 100. If the user is qualified, the new user is sent a form page such as illustrated in Figure 5 to initiate a real-time on-line registration 224. The form may, for example, require personal information and credit references from the user. The browser is able to transmit the data entered by the user in the blanks 502 as a "POST" message to the authentication server. A POST message causes form contents to be sent to the server in a data body other than as part of the URL. If the registration form filled out by the new user is valid 226, an appropriate SID is generated 228. If the registration is not valid, access is again denied 222.

An SID for an authorized user is appended ("tagged") 230 to the original [illegible] led [illegible]e on the content server. The authentication server then transmits a REDIRECT response 232 based on the tagged URL to the client browser 100. The modified URL, such as "http://content.com/[SID]/report" is automatically forwarded to the content server 120.

Figure 3, illustrates a typical client-server exchang involving the access control and monitoring method of the present invention. In Step 1, the client 50 running a browser transmits a GET request through a network for an uncontrolled page (UCP). For example, the user may request an advertisement page by transmitting a URL "http://content.com/advertisement", where "content.com" is the server name and "advertisement" is the uncontrolled page name. In Step 2, the content server 52 processes the GET request and transmits the requested page, "advertisement". The content server also logs the GET request in the transaction database 56 by recording the URL, the client IP address, and the current time.

In Step 3, the user on the client machine may elect to traverse a link in the advertisement page directed to a controlled page (CP). For example, the advertisement page

the client. In Step 9, the tagged URL is automatically forwarded by the browser as a GET request to the content server. The content server logs the GET request in the Transaction database 56 by recording the tagged URL, the client IP address, and the current time. In Step 10, the content server, upon validating the SID, transmits the requested controlled page "report" for display on the client browser.

According to one aspect of the present invention, the content server periodically evaluates the record contained in the transaction log 56 to determine the frequency and duration of accesses to the associated content server. The server counts requests to particular pages exclusive of [illegible]ated requests from a common client in order to determine the merits of the information on different pages for ratings purposes. By excluding repeated calls, the system avoids distortions by users attempting to "stuff the ballot box." In one embodiment, the time intervals between repeated requests by a [illegible]e measured to exclude those requests falling within [illegible]defined period of time.

Additionally, the server may, at any given time, track access history within a client-server session. Such a history profile informs the service provider about link transversal frequencies and link paths followed by users. This profile is produced by filtering transaction logs from one or more servers to select only transactions involving a particular user ID (UID). Two subsequent entries, A and B, corresponding to requests from a given user in these logs represent a link traversal from document A to document B made by the user in question. This information may be used to identify the most popular links to a specific page and to suggest where to insert new links to provide more direct access. In another embodiment, the access history is evaluated to determine traversed links leading to a

identification is most appropriately embedded in the session identifier described above.

In another aspect of the invention, facilities are provided to allow users to utilize conventional telephone numbers or other identifiers to access merchant services. These merchant services can optionally be protected using SIDs. In a preferred embodiment, as shown in Figure 6, a Web browser client 601 provides a "dial" command to accept a telephone number from a user, as by clicking on a "dial" icon and inputting the telephone number through the keyboard. The browser then constructs a URL of the form "http://directory.net/NUMBER", where NUMBER is the telephone number or other identifier specified by the user. The browser then performs a GET of the document specified by this URL, and contacts directory server 602, sending the NUMBER requested in Message 1.

In another embodiment [illegible] browser, client 601 uses a form page provided by directory server 601 that prompts for a telephone number or other identifier in place of a "dial" command, and Message 1 is a POST message to a URL specified by this form page.

Once NUMBER is received by directory server 601, the directory server uses database 604 to translate the NUMBER to a target URL that describes the merchant server and document that implements the service corresponding to NUMBER. This translation can ignore the punctuation of the number, therefore embedded parenthesis or dashes are not significant.

In another embodiment an identifier other than a number may be provided. For example, a user may enter a company name or product name without exact spelling. In such a case a "soundex" or other phonetic mapping can be used to permit words that sound alike to map to th same target URL. Multiple identifiers can also be used, such as

would first authenticate the user as belonging to the gold users group, and then would provide access to the "priority gold" page. An unpublished "ambassador" number could be directed to a tagged URL that permits access to the "priority gold" page without user authentication.

This invention has particular application to network sales systems such as presented in U.S. Patent Application Serial No. 08/328,133, filed October 24, 1994, by Payne et al. which is incorporated herein by reference.

Equivalents:

Those skilled in the art will know, or be able to ascertain using no more than routine experimentation, many equivalents to the specific embodiments or the invention described herein. These and all other equivalents are intended to be encompassed by the following claims.

```
/*
 *----------------------------------------------------------------------
 *
 * compute_ihash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 * a 32b xor of the 4 hash longwords.
 *
 * Results:
 *      hash int.
 *
 * Side effects:
 *      None.
 *----------------------------------------------------------------------
 */
int compute_ihash(char *str)
{
    MD5_CTX md5;
    unsigned char hash[16];
    unsigned int *p1;
    unsigned int hashi = 0;

    MD5Init(&md5);
    MD5Update(&md5, str, strlen(str));
    MD5Final(hash, &md5);
    p1 =    (unsigned int *) hash;

    hashi = *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    return hashi;
}

/*
 * ticket.c --
 *
 *      Commands for TICKET.
 *
 * Copyright 1995 by Open Market, Inc.
 * All rights reserved.
 *
 * This file contains pr prietary and confidential information and
 * remains the unpublished property of Open Market, Inc. Use,
 * disclosure, or reproduction is prohibited except as permitted by
 * express written license agreement with Open Market, Inc.
 *
```

```
   static char *GetAsciiDomain(char *domname, char *dflt);
   static int   computer_ihash(char *str);
   static char *computerHash(char *str);
   static char *GetSecret(int kid);
   static int   GetKidByKeyID(char *keyID);
   static char *CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid,
                          int exp, int uctx);
   static void freeTicketReqData(void *dataPtr);
   static void DumpStatus(HTTP_Request *reqPtr);
   static void TICKET_DebugHooks(ClientData clientData, char *suffix,
                          HTTP_Request *reqPtr);
   static int ParseSid(HTTP_Request *reqPtr);
   static int ParseTicket(HTTP_Request *reqPtr);
   static char *fieldParse(char *str, char sep, char **endptr);
   void TICKET_ConfigCheck();
   void DumpRusage(HTTP_Request *reqPtr);

/*
 *----------------------------------------------------------------------
 *
 * TICKET_RequireSidCmd --
 *
 *      Checks that the requested URL is authorized via SID to access this
 *      region. If the access is not authorized and we do not have a "remote"
 *      authentication server" registered, then an "unauthroized message"
 *      is returned.  If a "remote authentication server" has been
 *      declared, we REDIRECT to that server, passing the requested URL and
 *      required domain's as arguments.
 *
 * Results:
 *      Normal Tcl result, or a REDIRECT request.
 *
 * Side effects:
 *      Either an "unauthorized access" message or a REDIRECT in case of
error.
 *
 *----------------------------------------------------------------------
 */
static int TICKET_RequireSidCmd(ClientData clientData, Tcl_Interp *interp,
                          int argc, char **argv)
   {
   if (TicketGlobalData(EnableSidEater)) return TCL_OK;
   return(ProcessRequires(clientData, interp,argc, argv, ticketSid));
   }

/*
*----------------------------------------------------------------------
*
```

```
    /* compare the requesting SID/Ticket<DOM> to authorized list of domains */
    /* a match OR any valid domain and a required domain of TicketFreeArea is
su
   for (i = 1; i < argc; i++)
     {
     required_dom = GetDomain(argv[i] -1);
     if (required_dom != -1)
       {
       if (firstLegalDom == -1) firstLegalDom = required_dom;
       if ( (ticketPtr->sidDom == required_dom) ||
            (ticketPtr->valid && (ticketPtr->sidDom != -1) &&
              (required_dom == TicketGlobalData(FreeArea))) ||
            ((ticketPtr->ticketDom == required_dom) &&
              (time(0) <= ticketPtr->ticketExp) &&
              ((DStringLength(&ticketPtr->ticketIP) == 0) ||
              (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr-
>r
          )
            {
            DStringFree(&targetUrl);
            DStri:  e(&escapeUrl);
            return TCL_OK;
            }
     }
   }

/* count the number of domain crossing that caused re-auth */
if ((flavor == t  id) && (ticketPtr->sidDom) != -1) IncTicketCounte  Cou

/* authorization failed, if this was a sid url, and local auth is enabled */
/* or this was an access to the free area */
/* insert a new sid in the url, and REDIRECT back to the client 8?
if (TicketGlobalData(EnableLocalAuth) ||
     ((firstLegalDom == TicketGlobalData(FreeArea))
     && (flavor == ticketSid) && (firstLegalDom != -1)))
   {
   if ((DStringLength(&reqPtr->url) != 0) &&
        (DStringValue(&reqPtr->url)[0] != '/'))
     {
     HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
     DStringFree(&targetUrl);
     DStringFree(&escapeUrl);
     if (!ticketPtr->valid)
       DStringFree(&ticketPtr->sid);
     return TCL_RETURN;
     }
   NewSid =    CreateSid(reqPtr,
```

```
    DStringAppend(&escapeUrl, "}", -1);
    EscapeUrl(&escapeUrl);
    DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
    DStringFree(&escapeUrl);
    HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
    IncTicketCounter(CountRemoteRedirects);
    DStringFree(&targetUrl);*
    if (!ticketPtr->valid)
      DStringFree(&ticketPtr->sid);
    return TCL_RETURN;
    }

    /* authorization failed, if this is a ticket access, decode the */
    /* reason and handl via a redirect to a handler, or punt a      */
    /* no access message */
    if ((flavor == ticketTicket) && (firstLegalDom != -1) && (ticketPtr->ticketD
      {
      /* check For IP address restrictions */
      if ((DStringLength(&ticketPtr->ticket IP) != 0) &&
          (DstringLength(&TicketGlobalData(TicketAdrHandler)) != 0) &&
          (strcmp(DStringVa e(&ticketPtr->ticketIP), DStringValue(&reqPtr->remo
       {
       DSt                    ringValue(&TicketGlobalData(TicketAdrHandle
        ringAppend(&targetUrl, DStringValue(&ticketPtr->fields), -1);
        tringAppend(&targetUrl, "&url=", -1);
      DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
      IncTicketCounter(CountTicketAddr);
      HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
      DStringFree(&targetUr
      return TCL_RETURN;


   /* check for expired tick
   if (time(0) > ticketPtr->ticketE: )
     {
     DStringAppend(&targetUrl,    DStringValue(&TicketGlobalData(TicketExpHandle
     DStringAppend(&targetUrl,    DStringValue(&ticketPtr->fields), -1);
     DStringAppend(&targetUrl, "&url=", -1);
     DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
     IncTicketCounter(CountExpiredTicket);/*

  HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
  DStringFree(&targetUrl);
  return TCL_RETURN;
    }
  }
```

```
/* no handler, punt a message */
HTTP_Error(reqPtr, FORBIDDEN, "access denied by Require ticket/sid region
co
IncTicketCounter(CountNoRedirects);
if (!ticketPtr->valid)
DStringFree(&ticketPtr->sid);
DStringFree(&targetUrl);
DStringFree(&escapeUrl);
return TCL_RETURN;
}

/*
*----------------------------------------------------------------------
*
* Get(Ascii)Domain --
*  These routine performs an ascii to binary domain name lookup,
*  indexed by 'key') from the server's domain name catalog.  Name/number
*  pair's are loaded into the catalog at configuration time with the
*  with the "Domain" configuration command. The Ascii version returns
*  a pointer to a character string that represents the domain number.
*  The non Ascii version returns an integer representing the domain number.
*
* Results:
*  Integer value of domain.  If no domain is available, returns deflt.
*
* Side effects:
*  None.
*
*----------------------------------------------------------------------
*/
static int GetDomain (char *domname, int deflt)
   {
   HashEntry *entryPtr;
   DString DomName;

   DStringInit(&DomName);
   DStringAppend(&DomName, domname, -1);
   strtolower(DStringValue(&DomName));

   entryPtr = FindHashEntry(&TicketServerData.Domains,
DStringValue(&DomName));
   DStringFree(&DomName);
   if (entryPtr == NULL) return deflt;
```

```
    return (int) GetHashValue(entryPtr);
    }
static char * GetAsciiDomain(char *domname, char *deflt)
  {
  HashEntry *entryPtr;
  static char buffer[64];
  DString DomName;

  DStringInit (&DomName);
  DStringAppend(DomName, domname, -1);
  strtolower(DStringValue(&DomName));

  entryPtr = FindHashEntry(&TicketServerData.Domains,
DStringValue(&DomName));
  DStringFree(&DomName);
  if (entryPtr == NULL) return deflt;
  sprintf(buffer, "%d", (int) GetH...alue(entryPtr));
  return buffer;
  }
/*
*----------------------------------------------------------------------
*
*  TICKET_InsertLocalSid --
*
* Given a URL, inspect it to see if it refers to the local server/port
*  if it does, and it does not already contain a SID, insert one if
  ...urrent request included one. Note, for port 80 access look
* for a match with and without the port specifier.
*
* Results:
* None.
*
* Side effects:
  A SID may be inserted into the URL.
*
*----------------------------------------------------------------------
*/
void TICKET_InsertLocalSid(HTTP_Request *reqPtr, DString *result)
  {
  HTTP_Server *serverPtr;
  TICKET_Request *ticketPtr;
  char tmp[32];
  DString pattern1;
```

```
    DString pattern2;
    DString tmp_url;
    DString *hitPattern * NULL;

    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr,
 TicketServerData.tic
    if (ticketPtr == NULL) return;
    serverPtr = reqPtr->serverPtr;

    DStringInit(&pattern1);
    DStringInit(&pattern2);
    DStringInit(&tmp_url);

    DStringAppend(&pattern1, "http://", -1);
    DStringAppend(&pattern1, DStringValue(&serverPtr->serverName), -1);
    DStringAppend(&pattern2, DStringValue (&pattern1), -1);
    sprintf(tmp, ":%d", serverPtr->server_port);
    DStringAppend(&pattern1, tmp, -1);

    if ((DStringLength(result) >= DStringLength(&pattern1)) &&
     (strncasecmp(DStringValue(&pattern1), DStringValue(result),
DStringLengt hitPattern = &pattern1;
else
if  ((serverPTR-->server_port == 80) &&
    (DStringLength(result) >= DStringLength(&pattern2)) &&
    (strncasecmp(DStringValue(&pattern2), DStringValue(result),
DStringLength hitPattern + &pattern2;

    if (hitPattern ! = NULL)
    {
    DStringAppend(&tmp_url, DStringValue(hitPattern), -1;
    DStringAppend(tmp_url, DStringValue(&ticketPtr->sid), -1);
    DStringAppend(&tmp url, &DStringValue(result)
[DStringLength(hitPattern)],
    DStringFree(result);

    DStringAppend(result, DStringValue(&tmp_url), -1):
    DStringFree(&tmp_url);
    }

   DStringFree(&pattern1);
   DStringFree(&pattern2);
   DStringFree(&tmp_url);
   }
/*
```

```
*----------------------------------------------------------------------
*
* CreateSid --
*
   This routine takes the passed arguments and creates a sid.
*
* Results:
*  A sid.
*
* Side effects:
*
*----------------------------------------------------------------------
*/
char * CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid, int exp,
int uctx)
   {
   int bsid[3] = {0,0,0};
   char temp_str[512];
   DString hash;
   int act_hash;
   static char sid[64];
   unsigned int expire_time;
   char *secret;
   char *hashp;
   char *esp;
   unsigned char *esp;
   unsigned int sda;
   int endian = 1;

   DStringInit(&hash);
   expire_time =time(0)+ exp;

   put_sid(dom_lw,       dom_pos,     dom_mask,    dom);
   put_sid(uid_lw,       uid_pos,     uid_mask,    uid);
   put_sid(kid_lw,       kid_pos,     kid_mask,    kid);
   put_sid(exp_lw,       esp_pos,     exp_mask,
(expire_time>>exp_shft_amt))
   put_sid(uctx_lw,      uctx_pos,    uctx_mask,   uctx);
   put_sid(rev_lw,       rev_pos,     rev_mask,    sid_rev_zero);

   secret = GetSecret(kid);
   ASSERT (secret ! = NULL);
   DStringAppend(&hash, secret, -1);
```

```
   DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1;
    sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
    DStringAppend(&hash, temp_str, -1);
    /* format of the hash string is %s%s%08x%08x",
secret,ip_addr,bsid[2[,bsid[1
   hashP = DStringValue (&hash);
   act_hash = compute_ihash(hashP);
   while (*hashP ! = 0) *hashP++ = 0;
   DStringFree(&hash);
/*  fix_endian(&act_hash, ecp, eda); */

    .put_sid(sig_lw, sig_pos, sig_mask, act_hash)

/*   fix_endian(&bsid[0], ecp, eda); */
     fix_endian(&bsid[1], ecp, eda);
     fix_endian*&bsid[2], ecp, eda);

#if (1 == 0
   DumpSid();
#endif

     cp = radix64encode_noslash((char *) bsid, 12);
     strcpy(sid, SID_prefix);
     strcat(sid, cp);
     free(cp);
     return(sid);
     }

 /*
  *----------------------------------------------------------------------
  *
  *   compute_hash --
  *
  *   Compute the MD5 hash for the specified string, returning the hash as
  *   a 32 b xor of the 4 hash longwords.
  *
  *  Results:
  hash int.
  *

  *  Side effects:
  None.
 *----------------------------------------------------------------------
```

```
 */
static int compute_ihash(char *str)
  {
   MD5_CTX md5;
   unsigned char hash[16];
   unsigned int *pl;
   unsigned int hashi = 0;

  MDInit(&md5);
  MDUpdate(&md5, (unsigned char *) str, strlen(str));
  MDFinal(hash, &md5);
  pl = (unsigned int *) hash;

  hashi = *pl++;
  hashi ^= *pl++;
  hashi ^= *pl++;
  hashi ^= *pl++;
  return hashi;
  }

/*
 *----------------------------------------------------------------------
 *
 *  computeHash --
 *
 *  Compute the MD5 hash for the specified string, returning a
 *  32 character hex string.
 *
 *  Results:
 *  Pointer to static hash string.
 *
 * Side Effects:
 *  None.
 *----------------------------------------------------------------------
 */
static char *computeHash(char *str)
{
  int i;
  MD5_CTX md5;
  unsigned char hash[16];
  static char hashstr[33];
  char *q;
```

```
  MD5Init(&md5);
  MD5Update(&md5, (unsigned char *) str, strlen(str));
  MD5Final(hash, &md5);
  q = hashstr;
  for(i=0; i<16; i++ {
   sprintf(q, "%02x", hash[i]);
   q += 2;
  }
  *q = '\0';
  return hashstr;
}

/*
 *----------------------------------------------------------------------
 *
 *  TICKET_ParseTicket --
 *  Called by dorequest, before any region commands or mount handlers
 *  have run.  We parse and handle incomeing sid's and tickets.
 *
 * Results:
 *  None.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------

 */
int TICKET_ParseTicket(HTTP_Request *reqPtr)
  {
  int status = HT_OK;

  IncTicketCounter(CountTotalUrl);

  status = ParseSid(reqPtr);
  if (TicketGlobalData(EnableTicket) && (status == HT_OK)) status =
ParseTicke return status;
  }

/*
*----------------------------------------------------------------------
*
* ParseSid --
*
```

```
 * Called by TICKET_ParseTicket, before any region commands or mount handle
 * have run.  We parse and handle incomeing sid's.
 *
 * Results:
 * None.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------
 */

int ParseSid(HTTP_Request *reqPtr)
   {
   TICKEKT_Request *ticketPtr;
   HTTP_Server *serverPtr;
   DString hash;
   Int i;
   char *cp, *cp1;
   int *bsid=NULL, act_hash;
   unsigned int cur_tim, tdif, exp_tim;
   char *secret;
   char temp_str[512];
   char *hashP;
   int sid_ok = 0;
   unsigned char *ecp;
   unsigned int eda;
   int endian = 1;
   int ip1,ip2,ip3,ip4;

/* fetch the server private ticket extension data */
/* note that this sets up a default ticket block for both SID's and Ticket a
serverPtr = reqPtr->serverPtr;
ticketPtr = (TICKET_Request *) HT_GetReqExtData (reqPtr, TicketServerData.tic
ASSERT (ticketPtr == NULL);

ticketPtr = (TICKET_Request *) Malloc(sizeof(TICKET_Request));
HT_AddReqExtData(reqPtr, TicketServerData.ticketExtensionId, ticketPtr, free
DStringInit(&ticketPtr->rawUrl);
DStringInit(&ticketPtr->sid);
DStringInit(&ticketPtr->fields);
DStringInit(&TicketPtr->signature);
DStringInit(&TicketPtr->ticketIP);
ticketPtr->valid      =  0;
ticketPtr->sidDom     =   -1;
ticketPtr->ticketDom  =  -1;
ticketPtr->ticketExp  =  -1;
ticketPtr->uid        =  0
```

```
TicketPtr->uctx        =  0;
sscanf(DStringValue(&reqPtr->remoteAddr), "%d.%d.%d.%d", &ip1, &ip2, &ip3, &
ticketPtr->uid = (((ip1+ip2)<<24) | ((ip3+ip4)<<16) | (rand() & OxFFFF));
ticketPtr->uctx  = 1;
/* we are done if sids are not enabled, or this url does not have a sid */
if (!(TicketGlobalData(EnableSid))) return HT_OK;
cp1 = DStringValue(&reqPtr->url;
if (strstr(cp1, SID prefix) ! = cp1)
  return HT_OK;
if (strlen(cp1) == sidLength)
  {
  DStringAppend(&reqPtr->url, "/", -1);
  DStringAppend(&reqPtr->path, "/", -1);
  cp1 = DStringValue(&reqPtr->url);
  }
cp = strchr(cp1+sizeof(SID_prefix),'/');
if ((cp - cp1) != sidLength)
  return HT_OK;
IncTicketCounter(CountSidUrl);

DStringInit(&hash);

/* if sid eater is enabled, rewrite the url without the sid, and reprocess t
if (TicketGlobalDat(EnableSidEater))
  {
  DStringAppend(&hash, DStringValue(&reqPtr->url), -1);
  DStringFree(reqPtr->url);
  DStringAppend(&reqPtr->url, DStringValue(&hash)&hash)+sidLength, -1);
  DStringTrunc(&hash, 0);
  DStringAppend(&hash, DStringValue(&reqPtr-.path), -1);
  DStringFree(&reqPtr->path);
  DStringAppend(&reqPtr->path, DStringValue(&hash)+sidLength, -1);
  DStringFree(&hash);
  IncTicketCounter(CountDiscardedSidUrl);
  return HT_OK;
  }

DStringAppend(&ticketPtr->sid,  DStringValue(&reqPtr->url), sidLength);

/* first convert the SID back to binary*/
i = DStringLength(&ticketPtr->sid)-3;
bsid = (int *) radix64decode_noslash(DStringValue(&ticketPtr->sid)+3, i, &i)
iif ((bsid == NULL) || (i !+12)) goto rtn_exit;

fix_endian(&bsid[0], ecp, eda);
fix_endian(&bsid[1], ecp, eda);
fix_endian(&bsid[2], ecp, eda);
```

```
/* check the SID version field */
if (get_sid(rev_lw,rev_pos,rev_mask) ! = sid_rev_zero) goto sid_bad;
if (get_sid(rsrvl_lw,rsrvl_pos,rsrvl_mask) ! = 0)) goto sid_bad;
if (get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask) !+ 0) goto sid_bad;

/* Get a pointer to the secret */
secret = GetSecret(get_sid(kid_lw,kid_pos,kid_mask));
if (secret == NULL) goto sid_bad;

/* hash the sid and check the signature*/
DStringAppend(&hash, secret, -1);

DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1);
sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
dstringAppend(&hash, temp_str, -1);
/* format of the hash string is %s%s%08x%08x", secret,ip_addr,bsid[2],bsid[1

act_hash = compute_ihash(hashP);
while (*hashP != 0) *hashP== 0;
fix_endian(&act_hash, ecp, eda);
if (act_hash != get_sid(sig_lw,sig_pos,sig_mask)) goto  sid_bad;

/* is is ok, may be expired, but good enough to id user  */
ticketPtr->uiid = get_sid(uid_llw,uid_pos,uid_mask);
ticketPtr->uctx = get_sid(uctx_lw,uctx_pos,uctx_mask);

/* do the SID experation processing*/
cur_tim = (time(0)>>exp_shft_amt) & exp_mask;
expp_tim = get_sid(exp_lw,exp_pos,exp_mask);
tdif = (exp_tim - cur_tim) & Oxffff;
if (tdif > 0X7fff)
  {
  IncTicketCounter(countExpSid);
  goto sid_exp;
  }

/* sid is fine, save the sid state, update the url's */
ticketPtr->sidDom = get_sid(dom_lw,dom_pos,dom_mask);
ticketPtr->valid = 1;
sid_ok = 1;
IncTicketCounter(CountValidSid);

sid_bad:
  if (!(sid_ok)) IncTicketCounter(CountInvalidSid);
sid_exp:
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->path), -1);
```

```
    DStringTrunc(&reqPtr->path, 0);
    DStringAppend(&reqPtr->path, DStringValue(&ticketPtr->rawUrl)+sidLength, -1)

  DStringTrunc(&ticketPtr->rawUrl, 0);
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->url), -1);
  DStringTrunc(&reqPtr->url, 0);
  DStringAppend(&reqPtr->url, DStringValue(&ticketPtr->rawUrl)+sidLength, -1);

  rtn_exit:
    DStringFree(&hash);
    if (bsid != NULL) free(bsid);
    return HT_OK;
    }

  /*
   *----------------------------------------------------------------------
   *
   * freeTicketReqData
   *
   *  This routine frees the storage used by ticket specific request
   *  data.
   *
   * Results:
   *  None.
   *
   * Side effects:
   *  Memory freed.
   *
   *----------------------------------------------------------------------
  */

  ;tatic void freeTicketReqData(void *dataPtr)
    {
    TICKET_Request *ticketPtr = dataPtr;
    DStringFree(&ticketPtr->rawUrl);
    DStringFree(&ticketPtr->sid);
    DStringFree(&ticketPtr->fields);
    DStringFree(&ticketPtr->signature);
    DStringFree(&ticketPtr->ticketIP);
    free(ticketPtr);
    }
  /*
   *----------------------------------------------------------------------
   *
   * GetSecret --
   *
   *  Given a binary keyID, returns an ascii secret from the
```

```
 *  secrets store.
 *  for untranslatable names, return NULL.
 *
 * Results:
 *  "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
*/

char *GetSecret(int kid)
  {
  HashEntry *entryPtr;

  entryPtr = FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);
  if(entryPtr == NULL) return NULL;
  return DString.Value((DString *)GetHashValue(entryPtr));
  }


/*
 *----------------------------------------------------------------------
 *
 * GetKidByKeyID --
 *
 *  Given an ascii  KeyID return the binary Key ID.
 *  for untranslatable names, return -1.
 *
 * Results:
 *  "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
*/

int GetKidByKeyID(char *keyID)
  {
  HashEntry *entryPtr;

  entryPtr = FindHashEntry(*&TicketServerData.KeyID< (void *) keyID);
  if(entryPtr == NULL) return -1;
  return (int) GetHashValue(entryPtr);
  }
```

```
/*
 *----------------------------------------------------------------------
 *
 * fieldParse --
 *
 *  Given a string, a separator character, extracts a field up to the
 *  separator into the result string.
 *  Does substitution on '%XX' sequences, and returns the pointer to the
 *  character beyond last character in '*endptr'.
 *
 * Results:
 *  Returns a malloc'ed string (caller must free), or NULL if an
 *  error occurred during processing (such as an invalid '%' sequence).
 *
 * Side effects:
 *  None.
 *
 *----------------------------------------------------------------------
*/
#define SIZE_INC 200
statiic char *fieldParse(char *str, char sep, char **endptr)
{
  char buf[3];
  char c;
  char *end, *data, *p;
  int maxlen, len;

  len = 0;
  maxlen = SIZE_INC;
  p = data = malloc(maxlen);

/*
 * Loop through string, until end of string or sep character.
 */
 while(*str && *str != sep) {

  if(*str == '%') {

   if(!isxdigit(str([1]) || !isxdigit(str[2])) {
        free(data);
        return NULL;
   }
   buf[0] = str [1];
   buf[1] = str [2];
   buf[2] = '\0';
   c = strtol(buf, &end, 16);
   str += 3;
```

```
    } else if(*str == '+') {
     c = ' ';
     str++;
    } else
     c = *str++;

    *p++ = c;
    len++;
    if(len >= maxlen) {
     maxlen += SIZE_INC;
     data = realloc(data, maxlen);
     p = data + len;
    }


    }
    *p++ = '\0';
    *endptr = str;
    return data;
   }
/*
 *----------------------------------------------------------------------
 *
 *  DomainNameCmd --
 *
 *          this routine, builds the ascii domain name
 *  to binary domain name maping structure for a numeric domain.
 *  Syntax is Domain number name1 name2 name3 ...name...name_last
 *  At least one name is required.  The number is decimal and
 *  can be any value except -1. -1 is reserved as a marker
 *  for untranslatable names.
 *
 *  Results:
 *  None.
 *
 *  Side effects:
 *  Commands are validate, and entries added to the map
 *
 *----------------------------------------------------------------------
 */
static int DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                         int argc, char **argv)
  {
  int new,i;
  HashEntry *entryPtr;
  int DomNumber;
  DString DomName;
```

```
if (argc <3)
   {
   Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
                "arguments, should be \"3\"",
        (char *) NULL);
   return TCL_ERROR;
   }

DStringInit (&DomName);

if (((sscanf(argv[1], "%d", &DomNumber) ! = 1 || (DomNumber == -1)))
   {
   Tcl_AppendResult(interp, argv[0], " directive: ",
             "Domain number must be an integer, and not equal to -1",
             ", value found was ",argv[1],
             (char *) NULL);
   return to TCL_ERROR;
   }
for (i = 2; i < argc; i++)
   {
   DStringFree (&DomName);
   DStringAppend(&DomName, argv[i], -1);
   strtolower(DString Value(&DomName));
   entryPtr = CreateHashEntry(&TicketServerData.Domains, DStringValue
     (&DomNam
   if (new == 0)
   {
  Tcl_AppendResult(interp, argv[0], " directive:

    "Duplicate domain name specified, '", argv[i], "'",
    (char *) NULL);
  return TCL_ERROR;
   }
   SetHashValue(entryPtr, DomNumber);
   }
DStringFree(&DomName);
return TCL_OK;
}

/*
 *----------------------------------------------------------------------
 *
 *   SecretsCmd --
 *
 *  A call to this routine, builds kid to secrets table
 *
 *  Results:
```

```
 *  None.
 *
 *  Side effects:
 *  Secrets are stored.
 *
 *----------------------------------------------------------------------
 */
static int SecretsCmd(ClientData clientDate, Tcl_Interp *interp,
                int argc, char **argv)
  {
  int newKid,newKeyID;
  HashEntry *entryPtrKid = NULL, *entryPtrKeyID = NULL;
  int Kid;
  DString *dsptrKid;

  if (argc ! = 4)
   {
 Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
       "arguments, should be \"4\" ",
       (char *) NULL);
 return TCL_ERROR;
 }

  if (sscanf(argv[2], "%d", &Kid) ! = 1)
 {
 Tcl_AppendResult(interp, argv[0],
       " directive:  KeyID must be an integer",
       ", value found was '", argv[2], "'",
       (char *) NULL);
 return TCL_ERROR;
 }

entryPtrKid = CreateHashEntry(&TicketServerData.Secret[...]) Kid, &n
if (strlen(argv[1]))
  entryPtrKeyID = CreateHashEntry(&TicketServerData.KeyID, (void *) argv[1],
if ((newKid == 0 || ((newKeyID == 0) && strlen(argv[1]))))
{
Tcl_AppendResult(interp, argv[0],
       " directive:  Duplicate Secret specified for KeyID '",
       argv[1],
       (char *) NULL);
return TCL_ERROR;
}
if (strlen(argv[1]))
{
dsptrKid = (DString *) malloc(sizeof(DString));
DStringInit(dsptrKid);
```

```
    DStringAppend(dsptrKid, argv[3], -1);

    SetHashValue(entryPtrKid, dsptrKid);
    }
    SetHashValue(entryPtrKeyID, Kid);
    return TCL_OK;
    }


/*
 *----------------------------------------------------------------------
 *
 *  TICKET_Initialize --
 *
 *  Calls all the necessary routines to initialize the ticket subsystem.
 *
 *  Results:
 *       None.
 *
 *  Side effects:
 *       Commands added to the region interpreter.
 *  SID "/@@" url catcher declared.
 *
 *----------------------------------------------------------------------
 */
int TICKET_Initialize(HTTP_Server &serverPtr, Tcl_Interp *interp)
   {
   TicketServerData.ticketExtensionId = HT_RegisterExtension(serverPtr,
   "ticket

   InitHashTable(&TicketServerData.SecretsKid, TCL_ONE_WORD_KEYS);
   InitHashTable(&TicketServerdata.KeyID,    TCL_STRING_KEYS);
   InitHashTable(&TicketServerData.Domains, TCL_STRING_KEYS);

   /* initialize Server ticket data */
   DStringInit(&TicketGlobalData(AuthServer));
   DStringInit(&TicketGlobalData(TicketExpHandler));
   DStringInit(&TicketGlobalData(TicketAdrHandler));
   TicketGlobalData(FreeArea)                = 0;
   TicketGlobalData(EnableLocalAuth)         = 0;
   TicketGlobalData(CurrentSecret)           = 0;
   TicketGlobalData(EnableSid)               = 0;
   TicketGlobalData(EnableTicket)            = 0;
   TicketGlobalData(EnableSidEater)          = 0;
   TicketGlobalData(LocalAuthExp)            = 60*30;

   /* ticket event counters */
```

```
    TicketGlobalData(CountTotalUrl)          = 0;
    TicketGlobalData(CountSidUrl)            = 0;
    TicketGlobalData(CountValidSid)          = 0;
    TicketGlobalData(CountExpSid)            = 0;
    TicketGlobalData(CountInvalidSid)        = 0;
    TicketGlobalData(CountCrossDomain)       = 0;
    TicketGlobalData(CountLocalredirects)    = 0;
    TicketGlobalData(CountRemoteRedirects)   = 0;
    TicketGlobalData(CountNoRedirects)       = 0;
    TicketGlobalData(CountDiscardedSidUrl)   = 0;

    /* Ticket related Config commands */
    Tcl_CreateCommand(interp, "Domain",                    DomainNameCmd,
              (ClientData) serverPtr, NULL);
    Tcl_CreateCommand(interp, "Secrets",           SecretsCmd,
              (ClientData) serverPtr, NULL);
    Tcl_CreateCommand(interp, "AuthenticationServer", CmdStringValue,
          (ClientData) &TicketGlobalData(AuthServer), NULL);
    Tcl_CreateCommand(interp, "TicketExpirationHandler", CmdStringValue,
     (ClientData) &TicketGlobalData(TicketExpHandler), NULL);
    Tcl_CreateCommand(interp, "TicketAddressHandler", CmdStringValue,
     (ClientData) &TicketGlobalData(TicketAdrHandler), NULL);
    Tcl_CreateCommand(interp, "FreeDomain",          CmdIntValue,
     (ClientData) &TicketGlobalData(FreeArea), NULL);
    Tcl_CreateCommand(interp, "EnableSidEater",      CmdIntValue,
     (ClientData) &TicketGlobalData(EnableSidEater), NULL);
    Tcl_CreateCommand(interp, "EnableSid",           CmdIntValue,
     (ClientData) &TicketGlobalData(EnableSid), NULL);
    Tcl_CreateCommand(interp, "EnableTicket",        CmdIntValue,
     (ClientData) &TicketGlobalData(EnableTicket), NULL);
    Tcl_CreateCommand(interp, "EnableLocalAuth",     CmdIntValue,
     (ClientData) &TicketGlobalData(EnableLocalAuth), NULL);
    Tcl_CreateCommand(interp, "CurrentSecret",       CmdIntValue,
     (ClientData) &TicketGlobalData(CurrentSecret), NULL);
    Tcl_CreateCommand(interp, "LocalAuthExp",        CmdIntValue,
     (ClientData) &TicketGlobalData(LocalAuthExp), NULL);

    HT_AddMounthandler(serverPtr, (ClientData) NULL, TICKET_DebugHooks,
     "/omiserver", NULL);

    return HT_OK;
    }


/*
 *----------------------------------------------------------------------
 *
```

```
 * TICKET_Shutdown --
 *
 *  Calls all the necessary routines to shutdown the ticket subsystem.
 *
 *  Results:
 *  None.
 *
 *  Side effects:
 *  Memory freed
 *
 *----------------------------------------------------------------------
 */

void TICKET_Shutdown (HTTP_Server *serverPtr)
   {
   HashEntry *entryPtr;
   HashSearch search;
   DString   *dstring;

   DStringFree(&TicketGlobalData(AuthServer));
   DStringFree(&TicketGlobalData(TicketExpHandler));
   DStringFree(&TicketGlobalData(TicketAdrHandler));

   entryPtr = FirstHashEntry(&TicketServerData.SecretsKid, &search);
   while (entryPtr ! = NULL)
    {
    dstring = GetHashValue(entryPtr);
    DStringFree(dstring);
    free(dstring);
    entryPtr = NextHashEntry&search);
    }
   DeleteHashTable(&TicketServerData.SecretsKid);
   DeleteHashtable(&TicketServerData.KeyID);
   DeleteHashTable(&TicketServerData.Domains);
   }
/*
 *
----------------------------------------------------------------------
-
 *
 * TICKET_AddRegion Commands --
 *
 *         Add TICKET region commands for authentication/authorization
decisions.
 *
 * Results:
           None.
```

```
 *
 * Side effects:
 *          Commands added to the region interpreter.
 *
 *----------------------------------------------------------------------
 */

void TICKET_AddRegion Commands (HTTP_Request *reqPtr, Tcl_Interp *interp)
      {
      Tcl_CreateCommand(interp, "RequireSID", TICKET_RequireSidCmd,
                        (ClientData) reqPtr, NULL);
      Tcl_CreateCommand(interp, "RequireTicket", TICKET_RequireTicketCmd,
                        (ClientData) reqPtr, NULL);
      }


/*
 *------ -- ---- ----------------------------------------------------------
 *
 * TICKET_GetCGIVariables --
 *
 *     Add TICKET CGI variables to the CGI variable table.
 *
 * Results:
 *          None.
 *
 * Side effects:
 *          Extends the CGI variable hash table.
 *
 *----------------------------------------------------------------------
 */

void TICKET GetCGIVariables(HTTP_Request *req)
      {
      TICKET_Request *ticketPtr = (TICKET_Request *)
HT_GetReqExtData(req.TicketS

      /*
       * If there's no extension data, then we're not doing a ticket.  Just
return
      */

      if (ticketPtr == NULL)
        return)\;
```

```
        if  (DStringLength(&ticketPtr->rawUrl) ! = 0)
             HT_AddCGIParameter(req, "TICKET_URL", DStringValue(&ticketPtr-
>rawUrl), FA
        if (DStringLength (&ticketPtr->sid) != 0)
             HT_AddCGIParameter(req, "TICKET_SID", DStringValue(&ticketPtr-
>sid), FALSE
        if (DStringLength(&ticketPtr->fields) != 0)
             HT_AddCGIParameter(req, "TICKET_FIELDS", DStringValue(&ticketPtr-
>fields).
        if (DStringLength(&ticketPtr->signature) != 0)
        HT-AddCGIParameter(req, "TICKET_SIGNATURE". DStringValue(&ticketPtr-
>signa
        }/*
 *----------------------------------------------------------------------
 *
 *TICKET_GetUrl
 *
 *          Return the orignal url (with sid)
 *
 * Results:
 *          The URL.
 *
 * Side effects:
           None.
 *
 *----------------------------------------------------------------------
 */
char * TICKET_GetUrl(HTTP_Request *reqPtr)
        {
        TICKET_Request *ticketPtr;

        ticketPtr = (TICKET_Request *)
             HT_GetReqExtData(reqPtr, TicketServerData.ticketExtensionId);
        if ((ticketPtr != NULL) &&
              (DStringLength(&ticketPtr->rawUrl) != 0))
             return DStringValue(&ticketPtr->rawUrl);
        else
             return DStringValue(&reqPtr->url);
        }


/*
 *----------------------------------------------------------------------
 *
 * TICKET_ConfigCheck
 *
 *          Perform late configuration checks
```

```
 *
 * Results:
 *
 *
 * Side effects:
 *         Possible message loged/printed, and program exit'd.
 *
 *----------------------------------------------------------------------
 */
void TICKET_ConfigCheck()
      {
      HashEntry *entryPtr;
      int kid;

      if ((TicketGlobalData(EnableSid) & -0x1) != 0)
           {
           LogMessage(LOG_ERR, "EnableSid ... 0 or 1");
           exit (0);
           }
      if (!(TicketGlobalData(EnableSid))) return;

      kid = TicketGlobalData(CurrentSecret);
      if (kid && kid_mask) != kid)
           {
           LogMessage(LOG-ERR; "CurrentSecret %d is invalid", kid);
           exit(0);
           }

      entryPtr = FindHashEntry(&TicketServe...ta.SecretsKid, (void *) kid);

      if(entryPtr == NULL)
           {
           LogMessage(LOG_ERR), "No secret defined for CurrentSecret %d", kid;
           exit(0);

      if ((TicketGlobalData(FreeArea) & -0x255) != 0)
           {
           LogMessage(LOG_ERR, "FreeArea must be between 0 and 255");
           exit(0);
           }

      if ((TicketGlobalData(EnableSidTicket) & -0x1) != 0)
           {
           LogMessage(LOG_ERR, "EnableSidTicket must be 0 or 1");
           exit (0);
           }
```

```
        if ((TicketGlobalData(EnableTicket) & -0x1) !- 0);
             {
             LogMessage(LOG_ERR, "EnableTicket must be 0 or 1");
             exit(0);
             }

        if ((TicketGlobalData(EnableLocalAuth) & -0x1) != 0)
             {
             LogMessage(LOG_ERR, "EnablLocalAuth must be 0 or 1");
             exit (0);
             }
        }
/*
 *----------------------------------------------------------------------------
 *
 * TICKET_DebugHooks
 *
 *     Check for debug hooks and execute if found.
 *
 * Results:
 *     None.
 *
 * Side Effects:
 *     None.
 *
*----------------------------------------------------------------------------
 */
 tatic void TICKET_DebugHooks(ClientData clientData, char *suffix,
                              HTTP_Request *reqPtr)
       {
       if(strcmp(suffix, "/ticketstatus") == 0)
         {
         DumpStatus(reqPtr);
         HT_FinishRequest(reqPtr);
         return;
        }
    HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
    HT_FinishRequest(reqPtr);
    return;
    }
/*
 *----------------------------------------------------------------------------
 *
 * DumpStatus --
 *
 *     Dump the server's ticket stat's
 *
```

```
 * Results:
 *     None.
 *
 * Side effects:
 *     None.
 *
 *----------------------------------------------------------------------
 */
#define BUFSIZE 1024
static void DumpStatus(HTTP_Request *reqPtr)
{
    HTTP_Server *serverPtr = reqPtr->serverPtr;
    char tmp[BUFSIZE], timeStr[BUFSIZE];
    struct utsname sysinfo;
    time_t uptime;
    int hours;

    HTTP_beginHeader(reqPtr, "200 OK);
    HTTP_SendHeader(reqPtr, "Content-type: text/html", NULL);
    HTTP_EndHeader(reqPtr);
    HTTP_Send(reqPtr, "<title>WebServer Ticket Status</title>",
                      "<h1>WebServer Ticket Status</h1>:, NULL);

    HTTP_Send(reqPtr, "<p><hr><p><h2>Ticket Log</h2>", "<p><pre>\n", NULL);

    sprintf(tmp, "    <b>%s:  </b>  n", "Number of access          ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "    <b>%s:  </b>  %d\n", "Number of SID URL's      ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Valid SID's    ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Expired SID's  ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Invalid SID's  ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of XDomain accesses ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of Local Redirects  ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of Remote Redirects ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of No Auth servers  ", Ticket

    HTTP_Send(reqPtr, tmp, "</pre>", NULL);

    uptime = time (NULL) - serverPtr->started;
    uname(&sysinfo);
```

```
    striftime(timeStr, BUFSIZE, "%A, %d-%b-%y %T",
        localtime(serverPtr->started));

    springf(tmp, "Server runing on <d>%s</b> (%s %s) port %d, has been up \
                    since %s.<p>", sysinfo.nodename, sysinfo.sysname,
                      sysinfo.release, serverPtr->server_port, timeStr);
    HTTP_Send(reqPtr, tmp, NULL);

    sprintf(tmp, "    <b>Number of connections:        </b> %d\n",
              serverPtr->numConnects);
    HTTP_Send(reqPtr, tmp, "<p><pre>\n", tmp, NULL);
    sprintf(tmp, "    <b>Number of HTTP requests:      </b> %d\n",
    HTTP_Send(reqPtr, tmp, "</pre><p>", NULL);

    hours = max(uptime / 3600, 1);
    sprintf(tmp, "This server is averaging <b>%d</b> requests per hour.<p>",
              serverPtr->numRequests/hours);
    HTTP_Send(reqPtr, tmp. NULL);

    DumpRusage(reqPtr);
/*    DumpConnections(reqPtr); */

    DNS_DumpStats(reqPtr);

    HTTP_Send(reqPtr, "<p><hr><address>", DStringValue(&ht_serverSoftware),
              "</address>\n", NULL);

    reqPtr->done = TRUE;

}
#undef BUFSIZE
```

CLAIMS

What is claimed is:

1. A method of processing service requests from a client to a server system through a network comprising:

forwarding a service request from the client to the server system;

returning a session identifier from the server system to the client; and

appending the session identifier to the request and subsequent service requests from the client to the server system within a session of requests.

2. A method as claimed in Claim 1 wherein the server system tracks an access history of sequences of service requests within the session of requests.

3. A method as claimed in Claim 2 wherein the server system tracks the access history to determine service requests leading to a purchase made within the session of requests.

4. A method as claimed in Claim 1 wherein the server system counts requests to particular services exclusive of repeated requests from a common client.

5. A method as claimed in Claim 1 wherein the server system maintains a database relating customer information to access patterns, the information including customer demographics.

6. A method as claimed in Claim 1 wherein the server system subjects the client to an authorization routine prior to issuing the session identifier and the session identifier is protected from forgery.

7. A method as claimed in Claim 6 wherein the server system comprises plural servers including an authentication server which provides session identifiers for service requests to multiple servers.

8. A method as claimed in Claim 7 wherein:

a client directs a service request to a first server which is to provide the requested service;

the first server checks the service request for a session identifier and only services a service request having a valid session identifier, and where the service request has no valid identifier:

the first server redirects the service request from the client to the authorization server;

the authorization server subjects the client to the authorization routine and issues the session identifier to be appended to the service request to the first server;

the client forwards the service request appended with the session identifier to the first server; and

the first server recognizes the session identifier and services the service request to the client; and

the client appends the session identifier to subsequent service requests to the server system and is serviced without further authorization.

9. A method as claimed in Claims 1 or 7 wherein the session identifier includes a user identifier.

10. A method as claimed in Claims 1 or 7 wherein the session identifier includes an expiration time for the session.

11. A method as claimed in Claim 7 wherein the session identifier provides access to a protected domain to which the session has access authorization.

12. A method as claimed in Claim 11 wherein the session identifier is modified for access to a different protected domain.

13. A method as claimed in Claim 7 wherein the session identifier provides a key identifier for key management.

14. A method as claimed in Claims 1 or 7 wherein the server system records information from the session identifier in a transaction log in the server system.

15. A method as claimed in Claims 1 or 7 wherein communications between the client and server system are according to hypertext transfer protocol and the session identifier is appended as part of a path name in a uniform resource locator.

16. A method as claimed in Claim 15 wherein the client modifies the path name of a current uniform resource locator using relative addressing and retains the session identifier portion of the path name unmodified for successive requests in the session.

17. A method as claimed in Claim 1 or 7 further comprising excluding requests made to information from the client within a defined period of time.

18. A method of processing service requests from a client to a server system through a network comprising:

responding to a request for a document received from the client through the network;

appending a session identifier, which includes a user identification, to the request; and

returning the requested document wherein the document is customized for a particular user based on the user identification of the session identifier.

19. A method of processing service request for a document received from a client through network in which the document has been purchased by a user comprising:

responding to a request for a document received from a client through the network in which the document has been purchased by the user;

appending an authorization identifier to the request; and

returning the requested document if the authorization identifier indicates that the user is authorized to access the document.

20. A method as claimed in Claim 19, wherein the authorization identifier is encoded within a session identifier which is appended to the request.

21. A method of processing service requests from a client to a server system through a network comprising:

responding to a request for a document received from a client through the network;

appending a user identifier to the request;

returning the requested document to the client, and;

charging the user identified in the identifier for access to the document.

22. A method as claimed in Claim 21, wherein a user identifier is encoded within a session identifier which is appended to the request.

23. A method of processing service requests from a client to a server system through a network comprising:

forwarding a service request from the client to the server system; and

appending a session identifier to the request and subsequent service requests from the client to the server system within a session of requests.

24. An information system on a network comprising:

means for receiving service requests from clients and for [illegible] another service request that includes a session identifier;

means for providing the session identifier in response to an initial service request in a session of requests; and

means for servicing service requests from a client which include the session identifier, the subsequent service request being processed in the session.

25. An information system as claimed in Claim 24 wherein the means for providing the session identifier is in a server system which services the requests.

26. An information system as claimed in Claim 23 further comprising an authorization routine for authorizing

the client prior to issuing the session identifier and means for protecting the session identifier from forgery.

27. An information system as claimed in Claim 24 further comprising a transaction log for recording information from the session identifier.

28. An information system as claimed in Claim 24 further comprising means for tracking access history of sequences of service requests within the session.

29. An information system as claimed in Claim 24 further comprising means for counting requests to particular services exclusive of repeated requests from a common client.

30. An information system as claimed in Claim 24 further comprising a database relating customer information to access patterns, the information including customer demographics.

31. An information system as claimed in Claim 25 wherein communications between the client and server system are according to hypertext transfer protocol and the session identifier is appended as part of a path name in a uniform resource locator.

32. An information server on a network comprising:

means for responding to requests for hypertext pages received from a client through the network by returning the requested hypertext pages to the client;

means for responding to further requests derived from links in the hypertext pages; and

means for tracking the further requests derived from a particular hypertext page.

33. An information server as claimed in Claim 32 wherein the requests include a common session identifier and the server tracks requests within a session of requests.

34. An information server as claimed in Claim 32 further comprising a data base relating customer demographics to access patterns.

35. A method of providing access to information pages from a client to a server system through a network comprising:

providing a telephone number at the client;

mapping the telephone number to a target page identifier using a translation database;

requesting information described by the page identifier from the server system; and

displaying a page identified by the page identifier at the client.

36. A method of providing access to information pages from a client to a server system through a network comprising:

providing a descriptor at the client;

mapping the descriptor to a target page identifier using a translation database;

requesting at the client information described by the page identifier from the server system without further user action; and

displaying a page identified by the page identifier at the client.

37. A method as claimed in Claims 35 or 36 wherein the translation database resides in the server system which returns a uniform resource loctor in a REDIRECT command to the client to cause the client to request the information using the uniform resource locator.

38. A method as claimed in Claim 36 wherein the descriptor comprises a telephone number.

39. A method as claimed in Claim 36 wherein the descriptor comprises a descriptive term.

40. A method as claimed in Claim 39 wherein the term includes a company name.

41. A method as claimed in Claim 39 wherein the term includes a product name.

42. A method as claimed in Claim 39 wherein the term is identified by phonetic mapping.

43. A method as claimed in Claims 35 or 38 wherein the target page identifier describes a controlled page.

44. A method as claimed in Claims 35 or 36 wherein the target page identifier is a uniform resource locator.

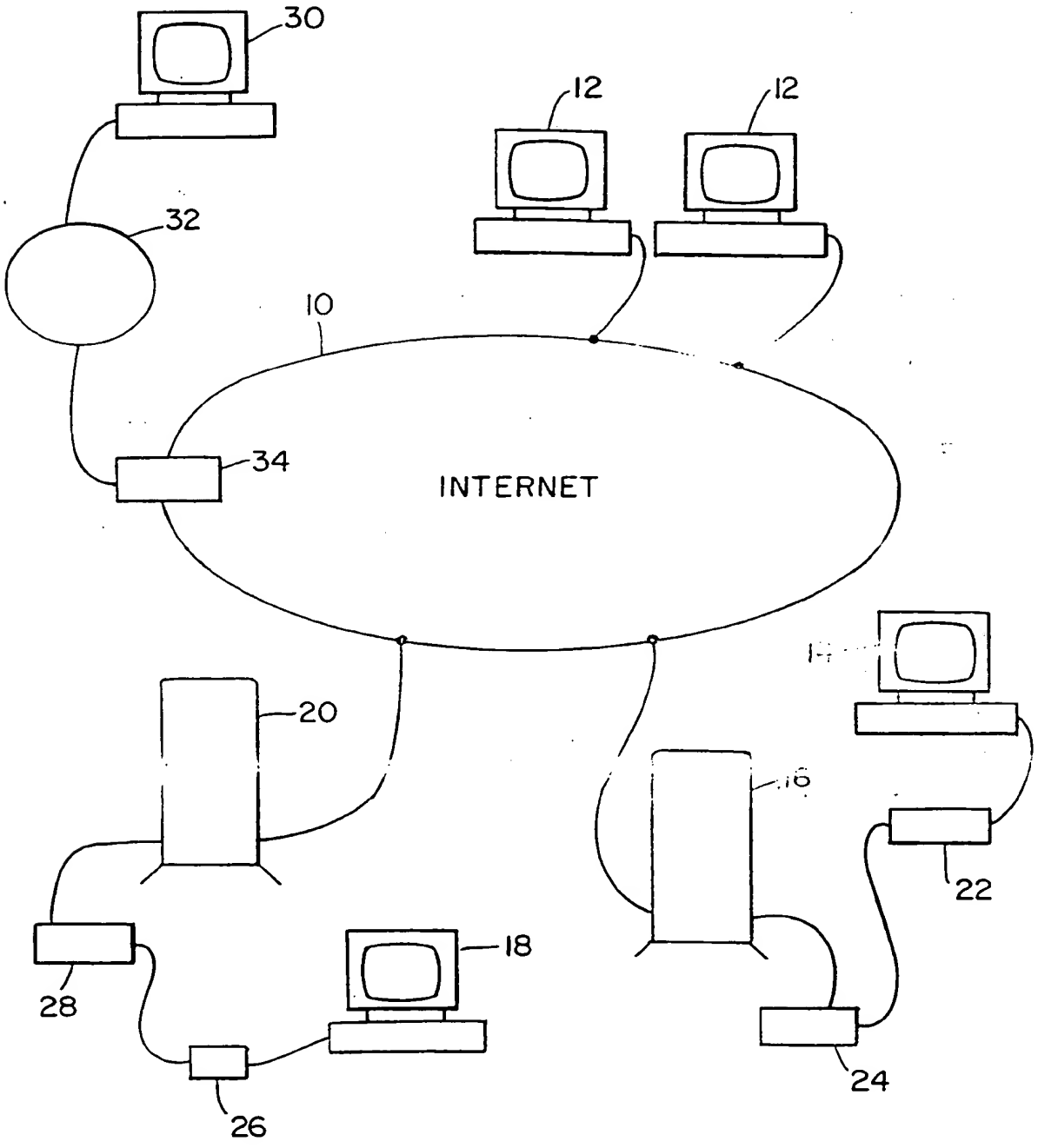


FIG. 1

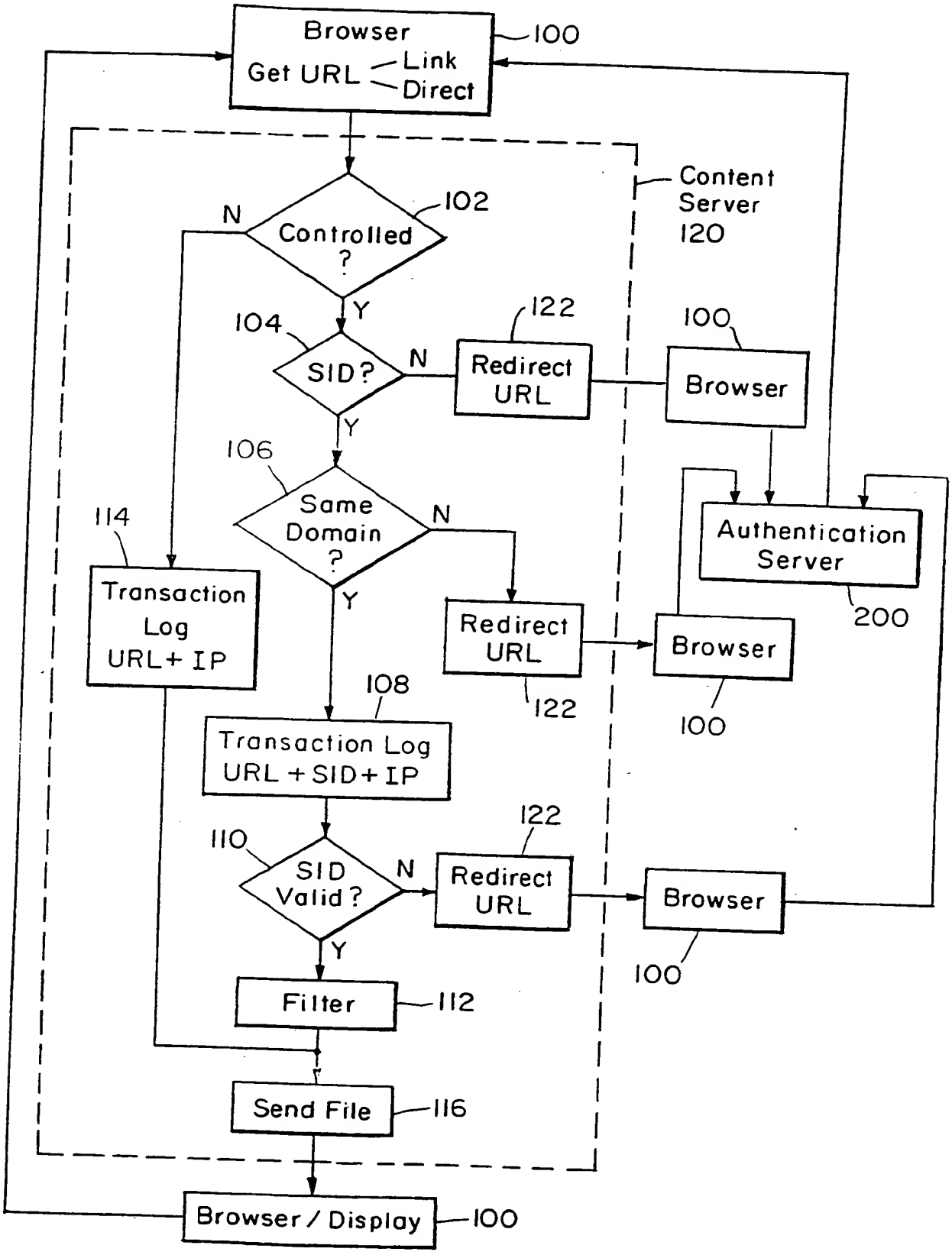

Browser
Get URL
Link
Direct
100
Content Server 120
Controlled ?
102
N
Y
SID?
104
N
Y
Redirect URL
122
Browser
100
Same Domain ?
106
N
Y
Transaction Log URL+IP
114
Authentication Server
200
Redirect URL
122
Browser
100
Transaction Log URL+SID+IP
108
SID Valid ?
110
N
Y
Redirect URL
122
Browser
100
Filter
112
Send File
116
Browser / Display
100

FIG. 2A

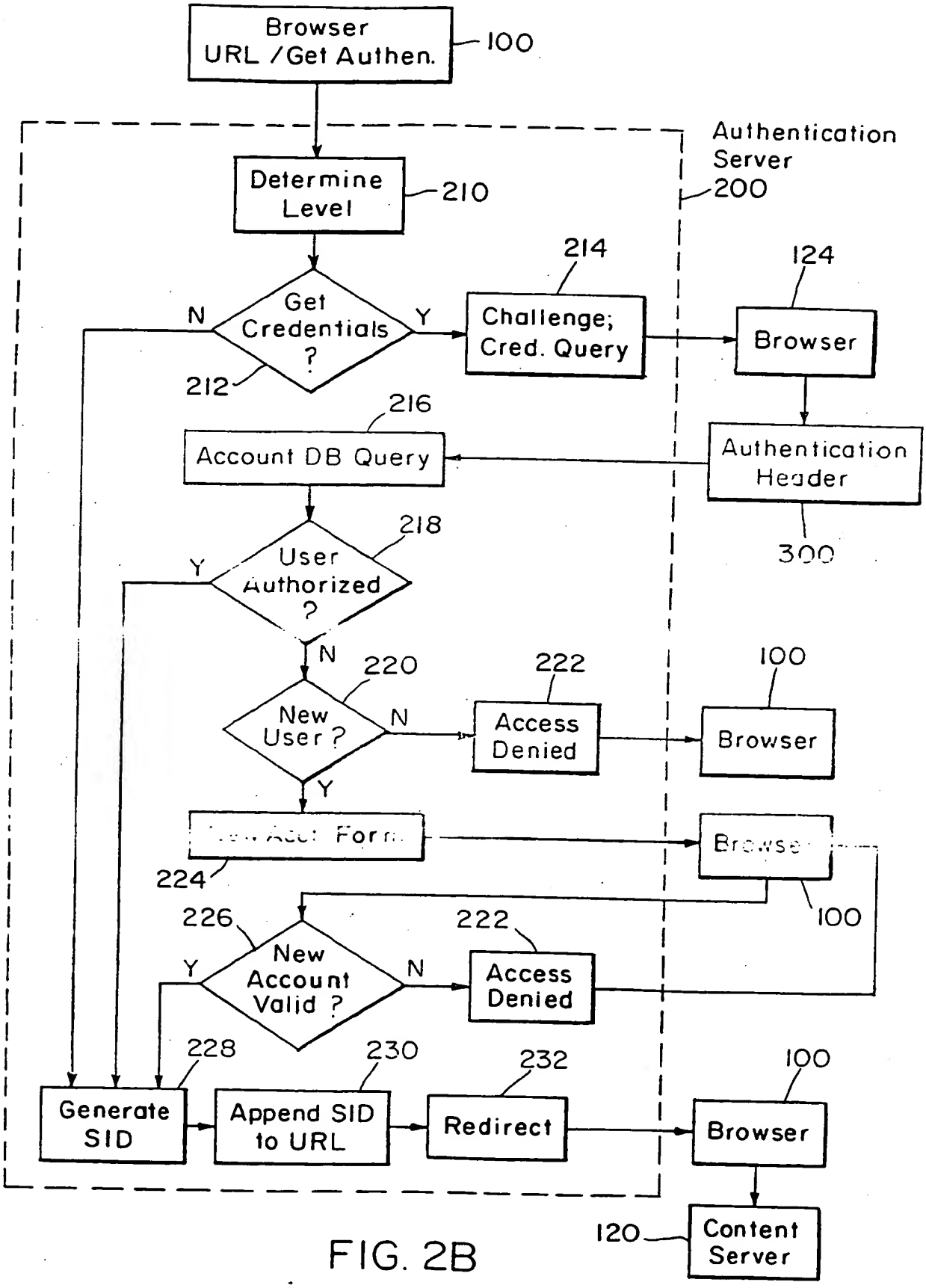

Browser URL /Get Authen.
100
Authentication Server
200
Determine Level
210
Get Credentials ?
212
N
Y
214
Challenge; Cred. Query
124
Browser
216
Account DB Query
Authentication Header
300
218
User Authorized ?
Y
N
220
New User ?
N
222
Access Denied
100
Browser
Y
224
Browse
100
226
New Account Valid ?
Y
N
222
Access Denied
228
Generate SID
230
Append SID to URL
232
Redirect
100
Browser
120
Content Server

FIG. 2B

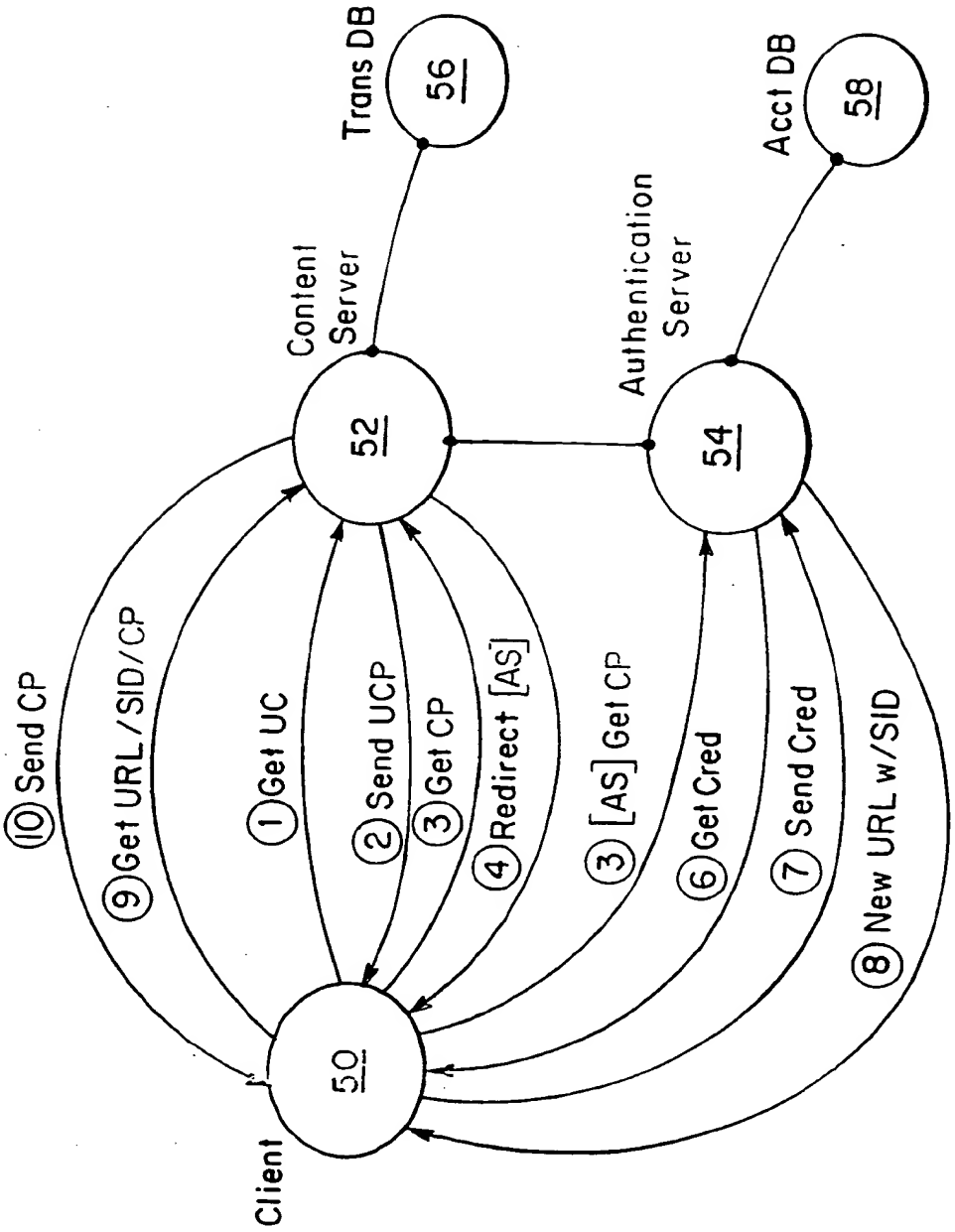

Trans DB
56
Acct DB
58
Content Server
52
Authentication Server
54
Client
50
① Get UC
② Send UCP
③ Get CP
④ Redirect [AS]
③ [AS] Get CP
⑥ Get Cred
⑦ Send Cred
⑧ New URL w/SID
⑨ Get URL / SID / CP
⑩ Send CP

FIG. 3

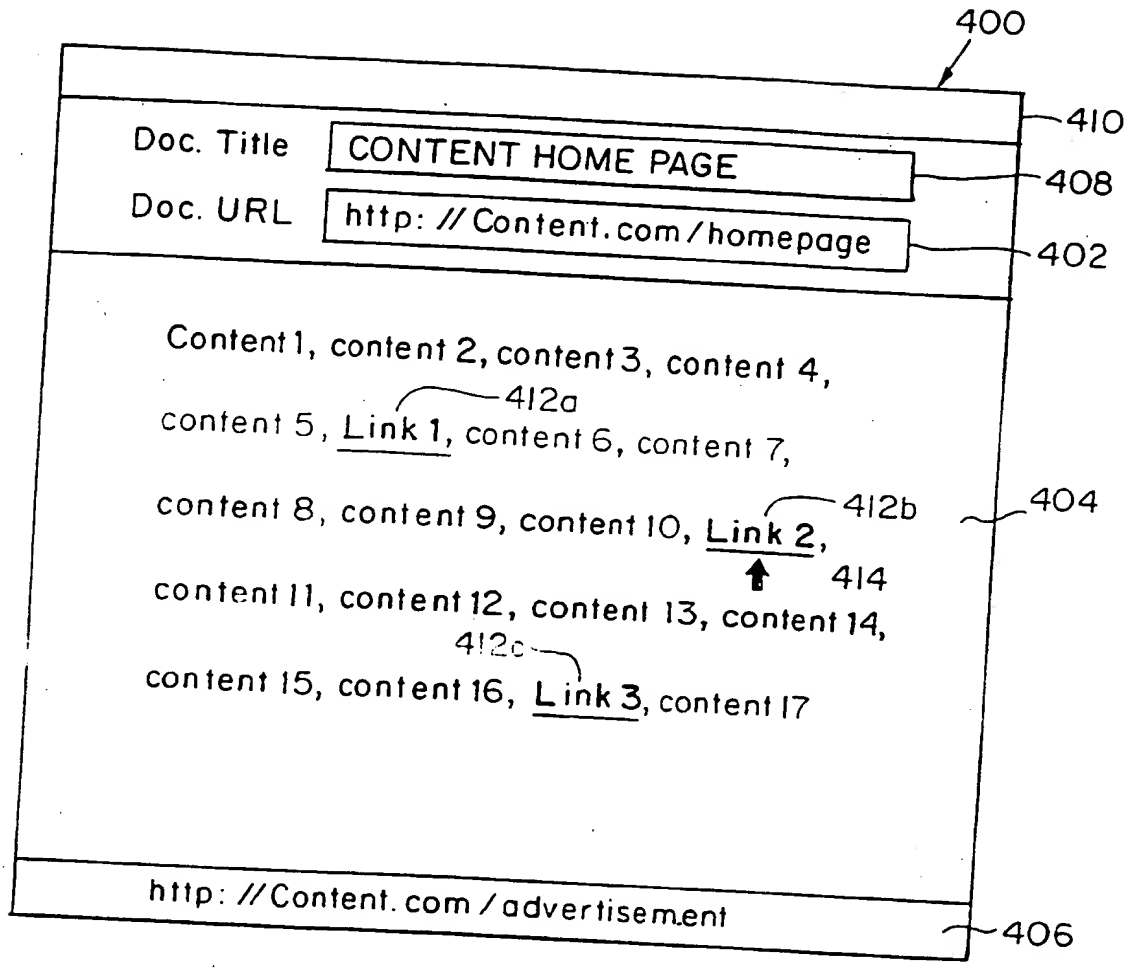

400
410
Doc. Title
CONTENT HOME PAGE
408
Doc. URL
http://Content.com/homepage
402
Content 1, content 2, content 3, content 4,
412a
content 5, Link 1, content 6, content 7,
content 8, content 9, content 10, Link 2,
412b
404
414
content 11, content 12, content 13, content 14,
412c
content 15, content 16, Link 3, content 17
http://Content.com/advertisement
406

FIG. 4

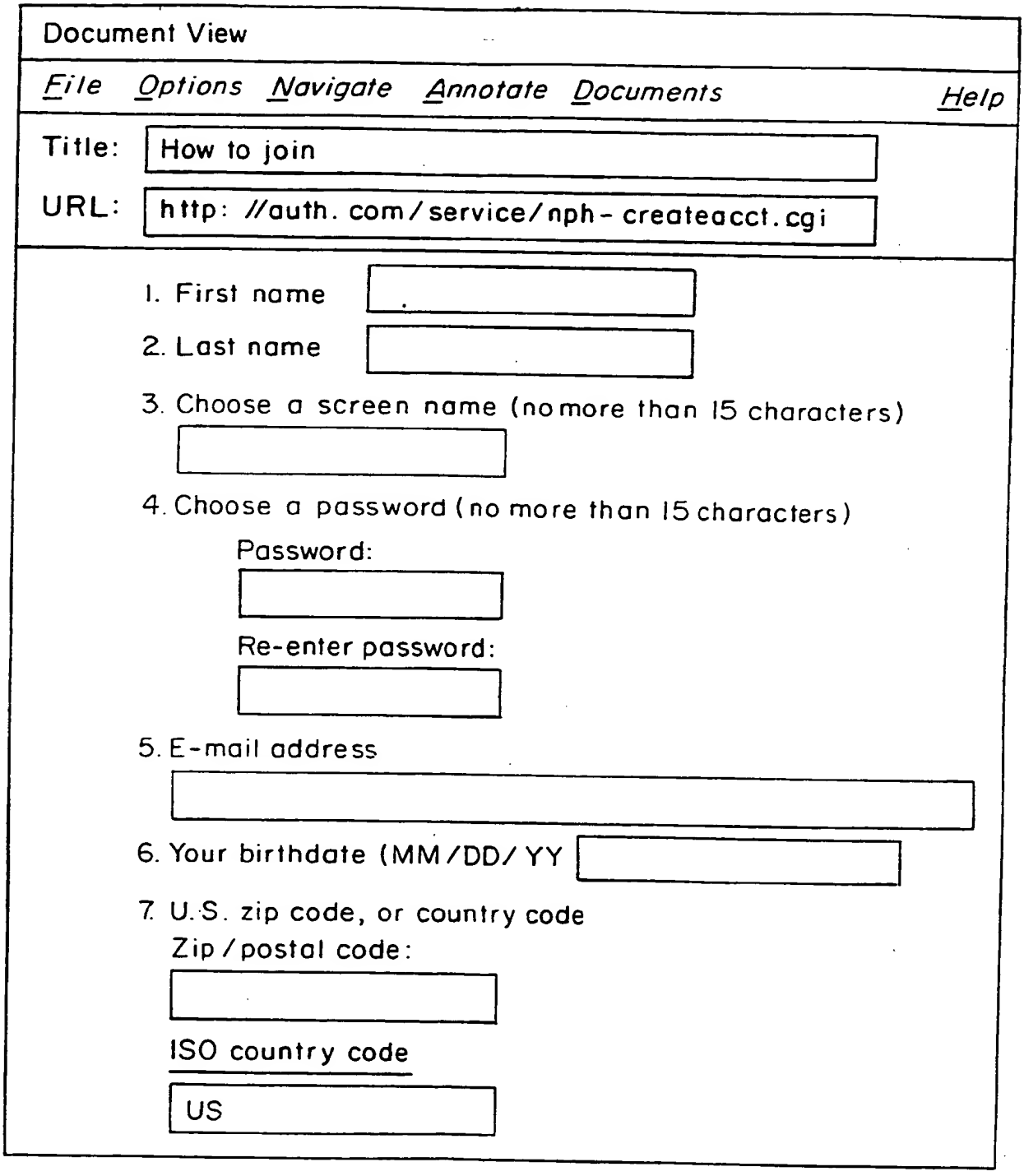

Document View
File Options Navigate Annotate Documents Help
Title: How to join
URL: http://auth.com/service/nph-createacct.cgi
1. First name
2. Last name
3. Choose a screen name (no more than 15 characters)
4. Choose a password (no more than 15 characters)
Password:
Re-enter password:
5. E-mail address
6. Your birthdate (MM/DD/YY
7. U.S. zip code, or country code
Zip/postal code:
ISO country code
US

FIG. 5

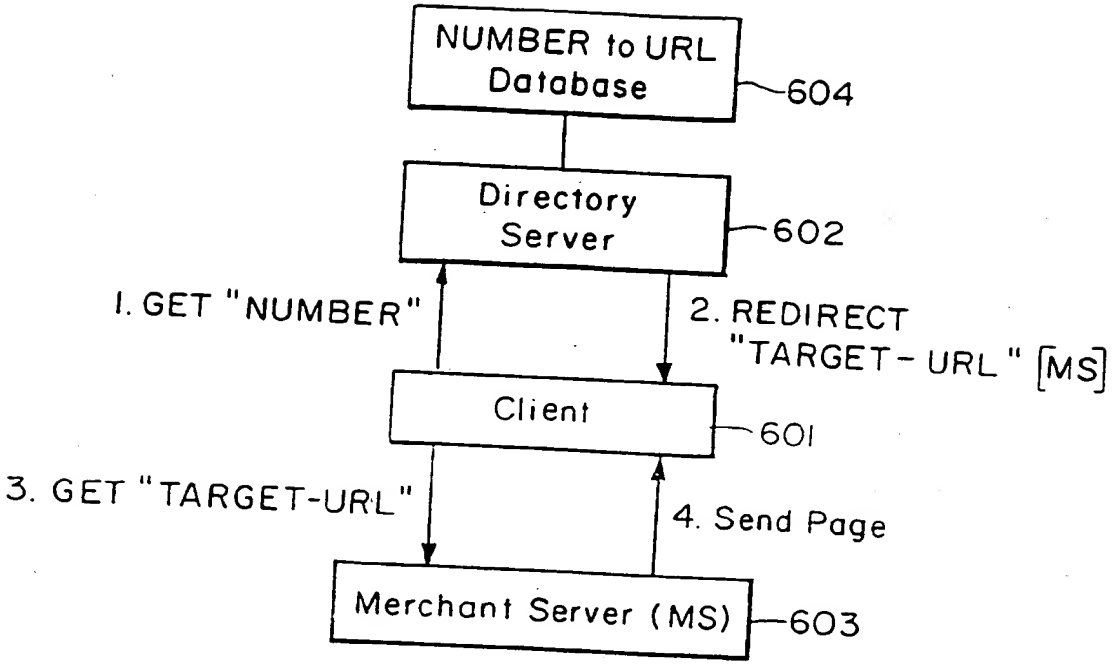


FIG. 6

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 96/07838

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y A	EP,A,0 456 920 (IBM) 21 November 1991 see page 5, line 33 - page 7, line 13 --- IEEE MULTIMEDIA, vol. 1, no. 1, 1994, COMPUTER SOCIETY US, pages 37-46, XP000440887 see page 43, left-hand column, line 12 - right-hand column, line 25 ---	1,10 2,4-8 3,9, 11-44 2,5
Y	EP,A,0 645 688 (KPN NEDERLAND) 29 March 1995 see column 1, line 37 - column 4, line 20 see figure 1 --- -/--	6-8

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
27 January 1997	04.02.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Canosa Areste, C

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 96/07838

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	IEEE NETWORK: THE MAGAZINE OF COMPUTER COMMUNICATIONS, vol. 9, no. 3, May 1995, NEW YORK US, pages 12-20, XP000505280 A.K.CHOUDHURY ET AL: "COPYRIGHT PROTECTION FOR ELECTRONIC PUBLISHING OVER COMPUTER NETWORKS" see page 14, right-hand column, line 49 - page 16, left-hand column, line 18 ---	4
A	WO,A,94 03959 (INTERNATIONAL STANDARD ELECTRIC CORP.) 17 February 1994 see page 5, line 25 - page 9, line 12 -----	1-44

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter nal Application No
PCT/US 96/07838

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-456920	21-11-91	US-A-	5560008	24-09-96
		JP-A-	3009444	17-01-91
		JP-B-	7060426	28-06-95
EP-A-645688	29-03-95	NL-A-	9301633	18-04-95
WO-A-9403959	17-02-94	DE-A-	4239754	03-02-94
		AU-A-	4703193	03-03-94
		DE-U-	9216139	18-02-93